14.2.2021

BILKENT UNIVERSITY

COMPUTER ENGINEERING DEPT.

CS202 HOMEWORK 1 SECTION 2

SORTING AND ALGORITHM EFFICIENCY

ZÜBEYİR BODUR

21702382

Spring 2021

1. Question 1

a) Part a

From the Big-O definition, if

$5n^3 + 4n^2 + 10 = O(n^4)$, then

$$5n^3 + 4n^2 + 10 \leq cn^4$$

for some c and n_0 when $n \geq n_0$. If we choose:

- $n_0 = 1$ and $c = 19$: $19 \leq 19$
- $n_0 = 2$ and $c = 5$: $66 \leq 80$
- $n_0 = 3$ and $c = 3$: $181 \leq 243$
- $n_0 = 4$ and $c = 1$: $373 \leq 512$

... and so on

b) Part b

i. Insertion Sort


8 24 51 28 20 29 21 17 38 27


8 24 51 28 20 29 21 17 38 27


8 24 28 51 20 29 21 17 38 27


8 24 28 51 20 29 21 17 38 27


8 24 28 51 20 29 21 17 38 27


8 24 28 20 51 29 21 17 38 27


8 24 20 28 51 29 21 17 38 27


8 20 24 28 51 29 21 17 38 27


8 20 24 28 51 29 21 17 38 27


8 20 24 28 51 29 21 17 38 27


8 20 24 28 29 50 21 17 38 27


8 20 24 28 29 51 21 17 38 27


8 20 24 28 29 51 21 17 38 27


8 20 24 28 29 21 51 17 38 27


8 20 24 28 21 29 51 17 38 27


8 20 24 21 28 29 51 17 38 27


8 20 21 24 28 29 51 17 38 27


8 20 21 24 28 29 51 17 38 27


8 20 21 24 28 29 51 17 38 27


8 20 21 24 28 29 17 51 38 27


8 20 21 24 28 17 29 51 38 27


8 20 21 24 17 28 29 51 38 27


8 20 21 17 24 28 29 51 38 27


8 20 17 21 24 28 29 51 38 27


8 17 20 21 24 28 29 51 38 27


8 17 20 21 24 28 29 51 38 27


8 17 20 21 24 28 29 51 38 27


8 17 20 21 24 28 29 38 51 27


8 17 20 21 24 28 29 38 51 27


8 17 20 21 24 28 29 38 51 27


8 17 20 21 24 28 29 38 27 51


8 17 20 21 24 28 29 27 38 51

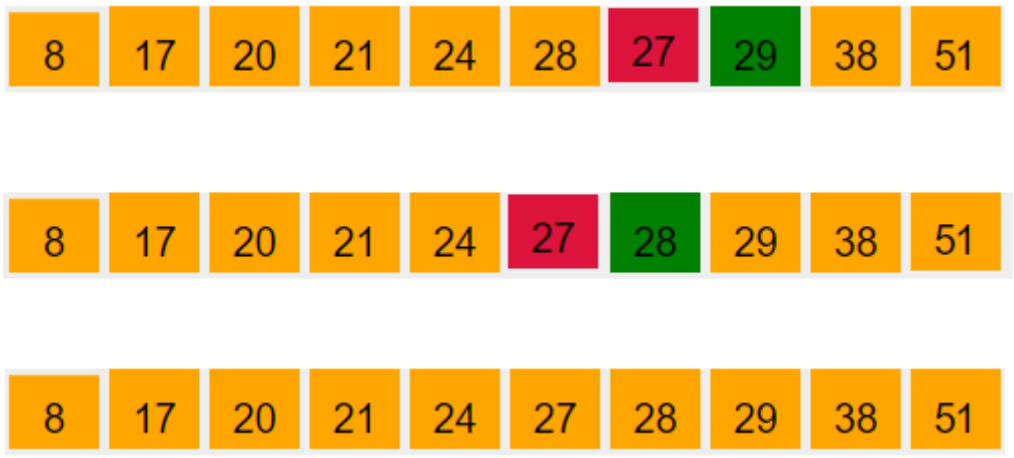
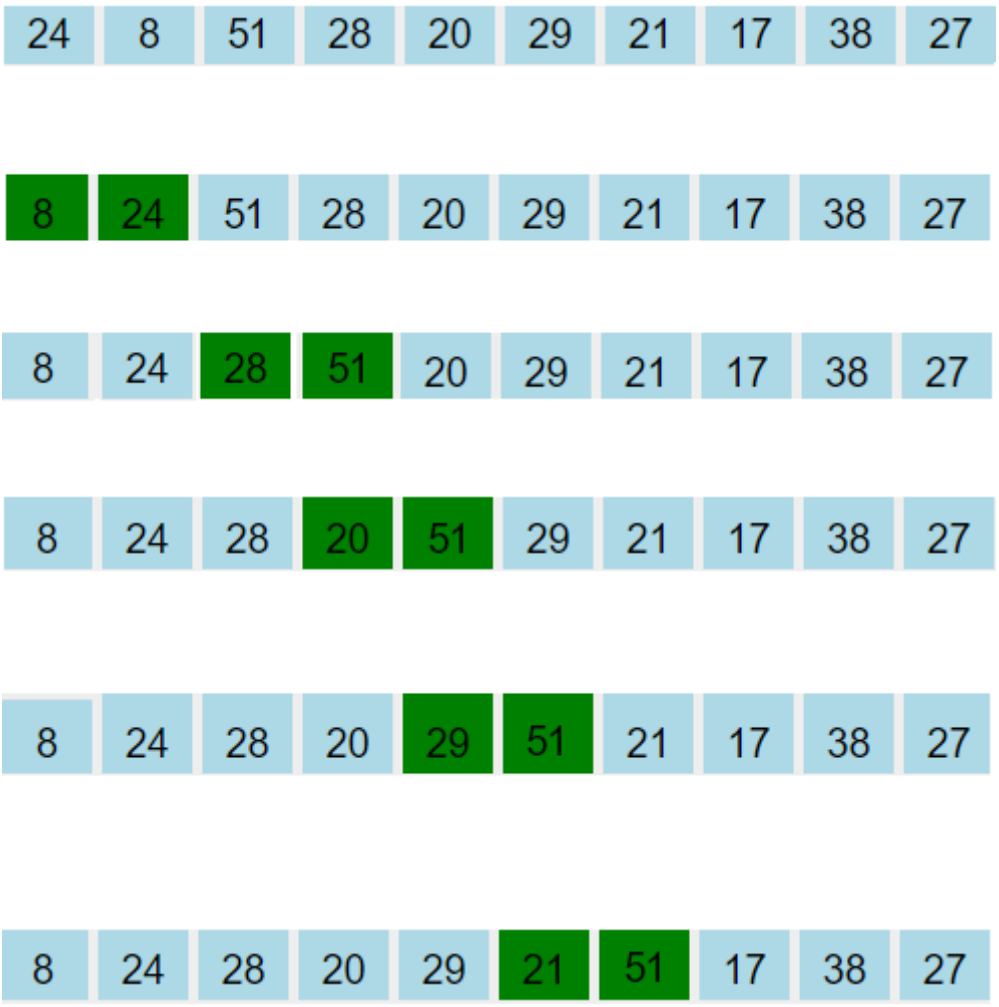


Figure 2: Algorithm tracing for insertion sort. All swap operations are shown.

ii. Bubble Sort




8 24 28 20 29 21 17 51 38 27


8 24 28 20 29 21 17 38 51 27


8 24 28 20 29 21 17 38 27 51

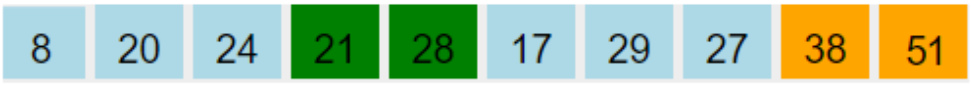

8 24 20 28 29 21 17 38 27 51


8 24 20 28 21 29 17 38 27 51

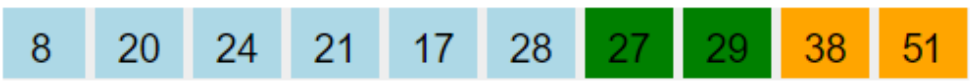

8 24 20 28 21 17 29 38 27 51


8 24 20 28 21 17 29 27 38 51


8 20 24 28 21 17 29 27 38 51


8 20 24 21 28 17 29 27 38 51


8 20 24 21 17 28 29 27 38 51


8 20 24 21 17 28 27 29 38 51

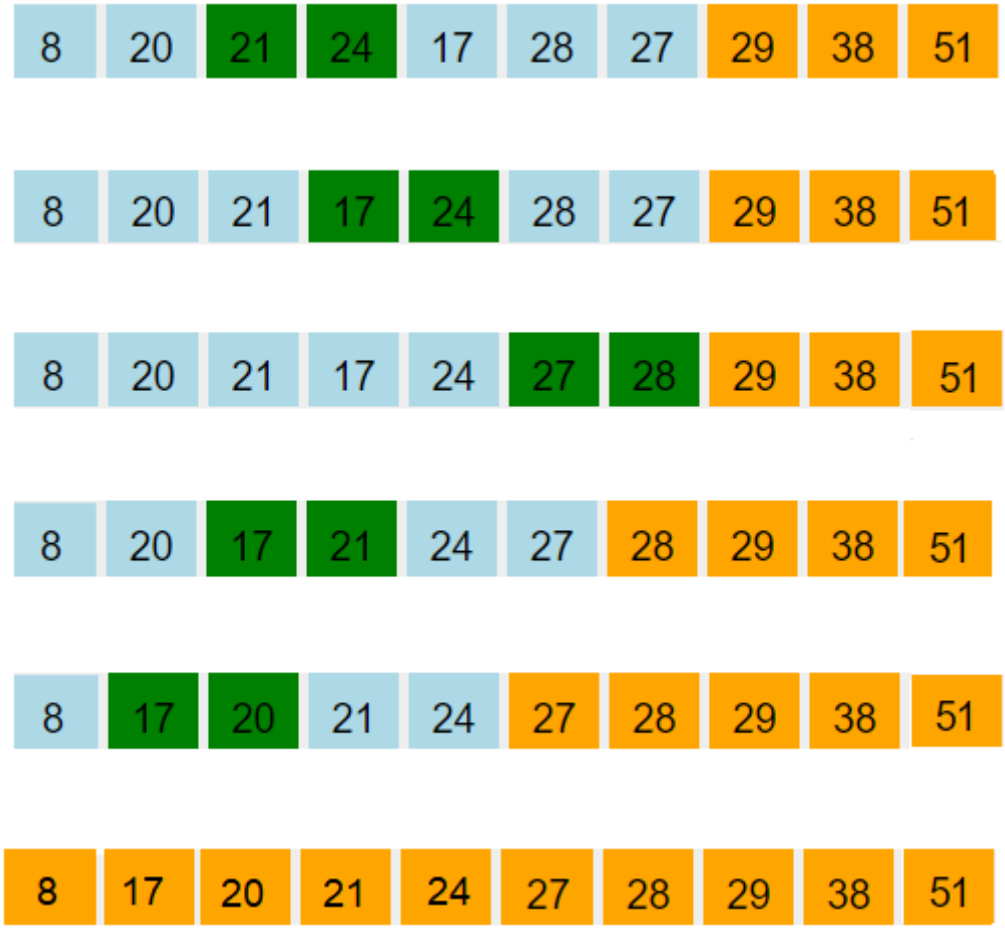


Figure 2: Algorithm tracing for bubble sort. Only bubbles containing swap operations are shown.

2. Output of the Program in Question 2

```
[zubeyir.bodur@dijkstra hw1]$ ./hw1
= INITIAL TESTING =

before sorting : [12, 7, 11, 18, 19, 9, 6, 14, 21, 3, 17, 20, 5, 12, 14, 8]
-------------------------------------------------------------------------------------
Analysis of Selection Sort
Array Size              compCount               moveCount
16                      120                     45
after sorting : [3, 5, 6, 7, 8, 9, 11, 12, 12, 14, 14, 17, 18, 19, 20, 21]

before sorting : [12, 7, 11, 18, 19, 9, 6, 14, 21, 3, 17, 20, 5, 12, 14, 8]
-------------------------------------------------------------------------------------
Analysis of Merge Sort
Array Size              compCount               moveCount
16                      46                      128
after sorting : [3, 5, 6, 7, 8, 9, 11, 12, 12, 14, 14, 17, 18, 19, 20, 21]

before sorting : [12, 7, 11, 18, 19, 9, 6, 14, 21, 3, 17, 20, 5, 12, 14, 8]
-------------------------------------------------------------------------------------
Analysis of Quick Sort
Array Size              compCount               moveCount
16                      45                      66
after sorting : [3, 5, 6, 7, 8, 9, 11, 12, 12, 14, 14, 17, 18, 19, 20, 21]

Analysis for Radix Sort
before sorting : [12, 7, 11, 18, 19, 9, 6, 14, 21, 3, 17, 20, 5, 12, 14, 8]
after sorting : [3, 5, 6, 7, 8, 9, 11, 12, 12, 14, 14, 17, 18, 19, 20, 21]
```

```
= EXPERIMENT FOR RANDOM ORDER =

Please wait for at least 10 seconds...
-------------------------------------------------------------------------------------
Analysis of Selection Sort
Array Size              Elapsed time (ms)       compCount               moveCount
6000                    90                      17997000                17997
10000                   250                     49995000                29997
14000                   490                     97993000                41997
18000                   810                     161991000               53997
22000                   1220                    241989000               65997
26000                   1700                    337987000               77997
30000                   2260                    449985000               89997
-------------------------------------------------------------------------------------
Analysis of Merge Sort
Array Size              Elapsed time (ms)       compCount               moveCount
6000                    2.04918                 67827                   151616
10000                   3.78788                 120545                  267232
14000                   5.49451                 175370                  387232
18000                   7.24638                 232044                  510464
22000                   8.92857                 290049                  638464
26000                   10.2041                 349302                  766464
30000                   12.1951                 408667                  894464
-------------------------------------------------------------------------------------
Analysis of Quick Sort
Array Size              Elapsed time (ms)       compCount               moveCount
6000                    1.51976                 87053                   135855
10000                   2.85714                 154417                  233724
14000                   3.90625                 214928                  315528
18000                   5.31915                 311293                  449340
22000                   6.66667                 364933                  583128
26000                   8.06452                 457887                  663411
30000                   9.43396                 530871                  824472
-------------------------------------------------------------------------------------
Analysis of Radix Sort
Array Size              Elapsed time (ms)
6000                    3.01205
10000                   5
14000                   7.04225
18000                   8.77193
22000                   10.6383
26000                   12.8205
30000                   15
```

Figure 3: Experiment for random order

```
= EXPERIMENT FOR ASCENDING ORDER =

Please wait for at least 10 seconds...
-------------------------------------------------------------------------------------------
Analysis of Selection Sort
Array Size              Elapsed time (ms)       compCount               moveCount
6000                    90                      17997000                17997
10000                   250                     49995000                29997
14000                   490                     97993000                41997
18000                   810                     161991000               53997
22000                   1210                    241989000               65997
26000                   1690                    337987000               77997
30000                   2260                    449985000               89997
-------------------------------------------------------------------------------------------
Analysis of Merge Sort
Array Size              Elapsed time (ms)       compCount               moveCount
6000                    1.3587                  39152                   151616
10000                   2.40385                 69008                   267232
14000                   3.52113                 99360                   387232
18000                   4.54545                 130592                  510464
22000                   5.68182                 165024                  638464
26000                   6.75676                 197072                  766464
30000                   7.69231                 227728                  894464
-------------------------------------------------------------------------------------------
Analysis of Quick Sort
Array Size              Elapsed time (ms)       compCount               moveCount
6000                    83.3333                 17997000                0
10000                   233.333                 49995000                0
14000                   455                     97993000                0
18000                   760                     161991000               0
22000                   1130                    241989000               0
26000                   1580                    337987000               0
30000                   2110                    449985000               0
-------------------------------------------------------------------------------------------
Analysis of Radix Sort
Array Size              Elapsed time (ms)
6000                    0.884956
10000                   1.92308
14000                   2.60417
18000                   3.40136
22000                   4.23729
26000                   4.80769
30000                   5.31915
```

Figure 4: Experiment for ascending order

```
= EXPERIMENT FOR DESCENDING ORDER =

Please wait for at least 10 seconds...
-------------------------------------------------------------------------------------
Analysis of Selection Sort
Array Size              Elapsed time (ms)       compCount               moveCount
6000                    90                      17997000                17997
10000                   250                     49995000                29997
14000                   490                     97993000                41997
18000                   810                     161991000               53997
22000                   1210                    241989000               65997
26000                   1690                    337987000               77997
30000                   2260                    449985000               89997
-------------------------------------------------------------------------------------
Analysis of Merge Sort
Array Size              Elapsed time (ms)       compCount               moveCount
6000                    1.32626                 36656                   151616
10000                   2.43902                 64608                   267232
14000                   3.52113                 94256                   387232
18000                   4.7619                  124640                  510464
22000                   5.61798                 154208                  638464
26000                   6.84932                 186160                  766464
30000                   7.8125                  219504                  894464
-------------------------------------------------------------------------------------
Analysis of Quick Sort
Array Size              Elapsed time (ms)       compCount               moveCount
6000                    166.667                 17997000                27000000
10000                   470                     49995000                75000000
14000                   920                     97993000                147000000
18000                   1520                    161991000               243000000
22000                   2280                    241989000               363000000
26000                   3180                    337987000               507000000
30000                   4230                    449985000               675000000
-------------------------------------------------------------------------------------
Analysis of Radix Sort
Array Size              Elapsed time (ms)
6000                    0.888099
10000                   1.89394
14000                   2.6455
18000                   3.40136
22000                   4.20168
26000                   4.9505
30000                   5.55556
```

Figure 5: Experiment for descending order

3. Question 3

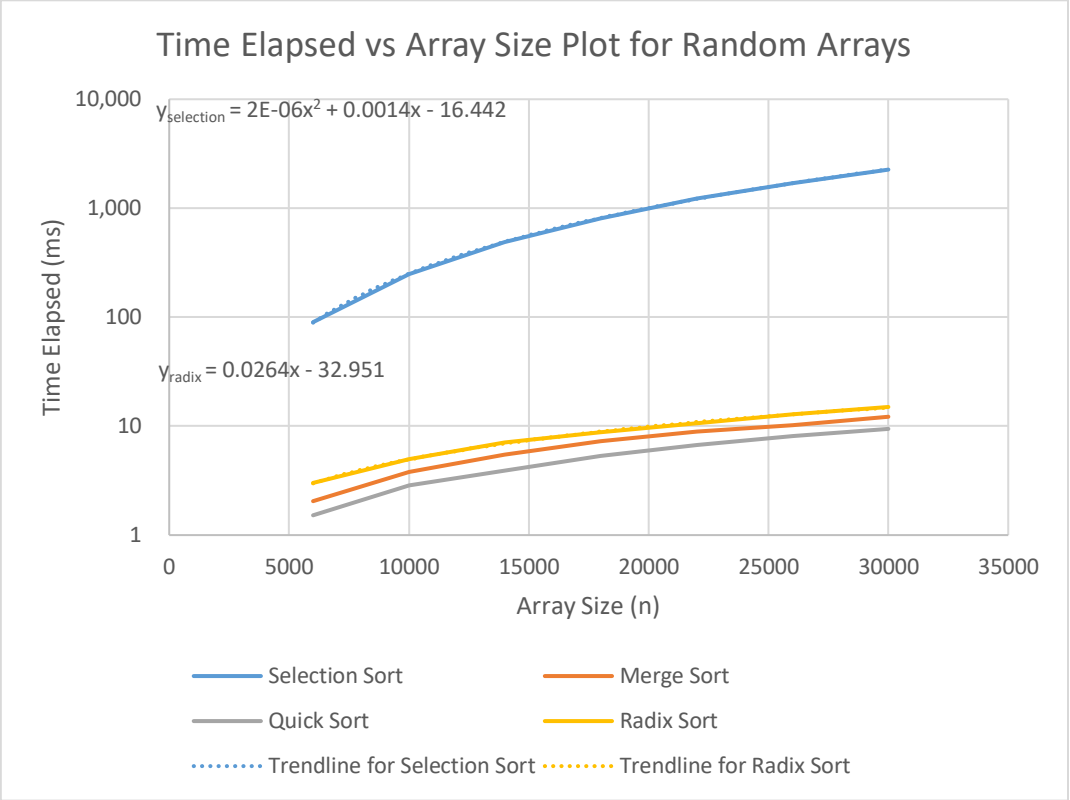


Figure 6: Plot of Time Elapsed vs Array Size for Arrays Containing Randomly Generated Numbers

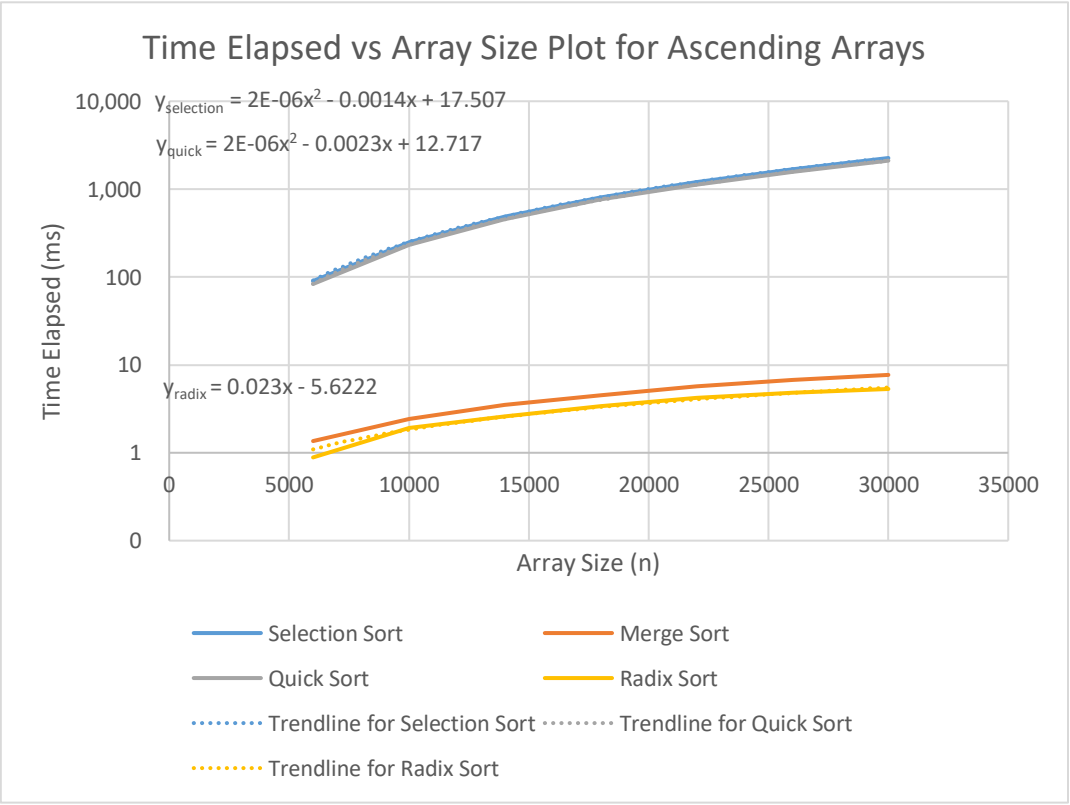


Figure 7: Plot of Time Elapsed vs Array Size for Sorted Arrays in Ascending Order

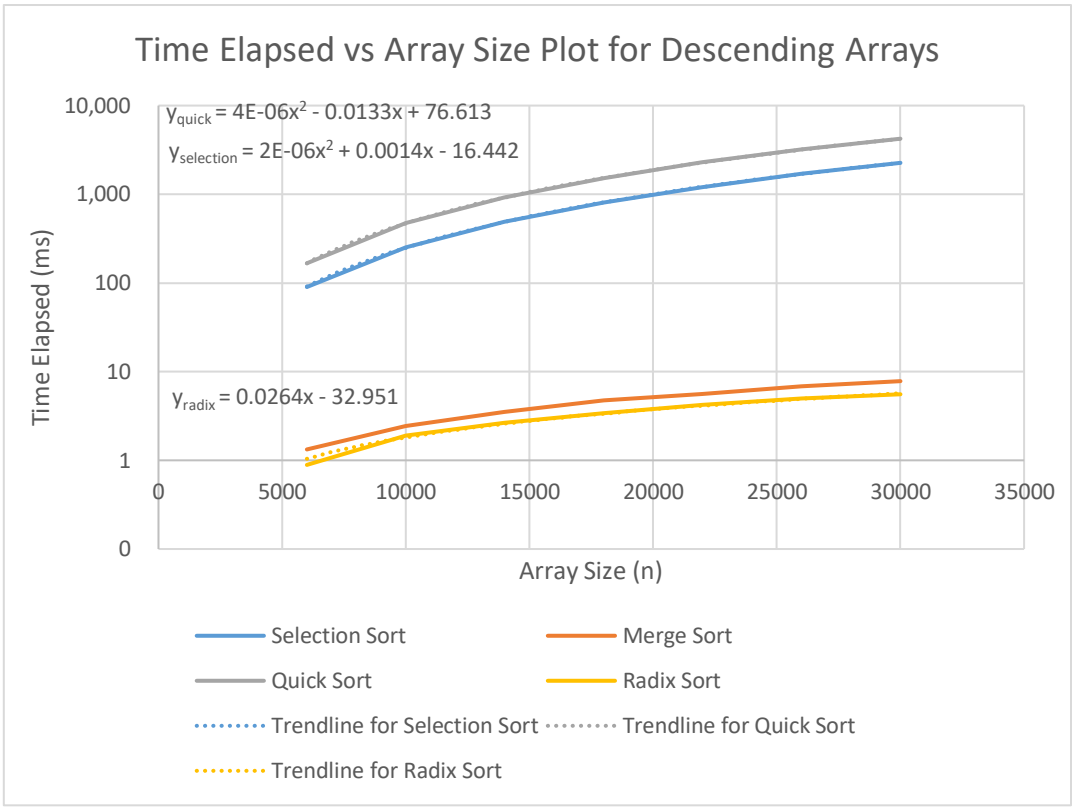


Figure 8: Plot of Time Elapsed vs Array Size for Sorted Arrays in Descending Order

From the trendlines generated from the experimental results, we can see that our theoretical efficiency of radix sort and selection sort is represented in same complexity in Big-O notation.

- Selection sort executed in exact same efficiency in three different experiments. We can see this from the output of the program. We can also observe that the experimental value for the time complexity of selection sort was also $\theta(n^2)$, as in theoretical value.

- Radix sort is $\theta(n)$ if d is constant. For the sorted arrays, d is constant as it's either 4 or 5. For the random arrays, the randomness of d didn't affect the outcome significantly. In fact, d has a maximum value as we are using int type for the arrays. In this case, maximum element is 2.147.483.647, meaning that d is maximum 10. Therefore, d has an upper bound of a constant number, meaning we can take d as a constant too. In fact, the output of the program and the graphs also tell us that radix sort

has liner time complexity. However, we can't conclude that radix sort is $\theta(nd)$ from this experiment.

- Quick sort is slightly more efficient than merge sort if the array is not sorted; though this difference is really small. For their experimental complexity, all we can say is that they are smaller than $\theta(n^2)$, and larger than $\theta(n)$, from the experiment for random arrays. This also holds for the theoretical average time complexity for both algorithms, which is $\theta(n \log n)$.

- Merge sort's efficiency is the same for sorted arrays and randomly generated arrays.

- Quick sort is as efficient as selection sort if the array is already sorted in ascending order. This happens because we choose the first item as the pivot, which is the worst case of quick sort, where the first item is the pivot and the array is already sorted. If the array is sorted in descending order, quick sort is even slower than selection sort.